**Machine Learning Analysis of Voting Tendencies by Electoral District:**

**Income-Based Voter Behavior in Washington State**

Yeonbi Han, Data Science Master Student

Graduate Center, CUNY

Spring 2024 Capstone Project

yhan4@gradcenter.cuny.edu

**Google Drive link(Voting model)**

https://drive.google.com/drive/folders/1pWSvWuceYee85gEGGSM0AXLl0G7rgDSk?usp=drive\_link

**Abstract**

This paper investigates the development and effectiveness of a machine learning model for analyzing voting patterns using income data from Washington state. By integrating voting data with geographic information, this study focuses on constructing a more exact voter behavior prediction model through batch processing, which combines various data sources. This research was conducted in collaboration with other team members, who analyzed different elements such as gender, age, and education in other states including Texas. This multi-faceted approach is expected to offer deeper insights into political voting patterns, offering valuable insights for policymakers and analysts.

**Introduction**

Understanding voter behavior patterns plays a crucial role in political decision-making and strategy formulation in modern society. Predicting and understanding voters' tendencies in upcoming elections is particularly crucial as it can significantly influence the direction of political campaigns. This research applies machine learning techniques to analyze voting patterns, aiming to develop a more precise model for predicting voter behavior.

Centered around Washington state and other states, this study incorporates various demographic variables such as income, education, gender, and age. Each variable differently influences voter behavior, hence, integrating these data helps a multifaceted understanding of complex voter actions.

This research has two primary goals: firstly, to develop a machine learning model that can more accurately predict voter behavior through the integration of various data sources. Secondly, to present an integrated model of voting pattern analysis that considers regional differences through the collaboration with team members who have collected data from different regions. This approach is expected to effectively interpret the complexities of voting patterns and contribute significantly to the formulation of strategies for upcoming elections.

**Related Work**

This paper explores the use of various machine learning techniques and data integration methods for predicting voter behavior and analyzing voting patterns. While earlier studies have primarily focused on single variables or limited data sources, this research integrates a wide range of demographic variables such as income, gender, age, and educational levels to conduct more precise analyses. This approach allows for a detailed examination of the impact of each variable on voter behavior and contributes significantly to the development of more accurate prediction models. Moreover, by reflecting the multidimensional characteristics of data, this study introduces new methodologies that enhance the accuracy of voting pattern predictions by analyzing the complex interactions between election data and voter behavior.

**Approach**

This code encompasses a comprehensive process of preparing data, training a deep learning model, and evaluating the results. Below, I will summarize each step and propose improvements for the modeling and evaluation approaches.

1. **Data Preparation and Preprocessing:** Data is loaded from a CSV file using pandas. Unnecessary features are removed, and data types are adjusted. There's also a need to explicitly specify data types based on warning messages (e.g., using the dtype option). The data is divided into training and testing sets.
2. **Model Design and Implementation:** Two models, SimpleNN and DeepNN, are implemented. DeepNN includes multiple layers with batch normalization and dropout, making it a more complex structure. A custom loss function is implemented to calculate MSE (Mean Squared Error).
3. **Training and Optimization:** The model is trained using SGD and Adam optimizers. The pros and cons of each optimizer are compared, and training losses are visualized.
4. **Evaluation and Validation:** The RMSE (Root Mean Squared Error) is calculated to assess the predictive performance of the model. Predictions are made on the test dataset, and losses over time are plotted.

These structured steps provide a robust framework for advancing the model's effectiveness through systematic data handling, iterative model enhancements, and rigorous validation.

**Preparation of Datasets**

In our study, we conducted a comprehensive review of several datasets to understand voter behavior patterns and their relationship with socio-economic factors. The datasets were obtained through several reputable sources, each serving a unique purpose in our analysis.

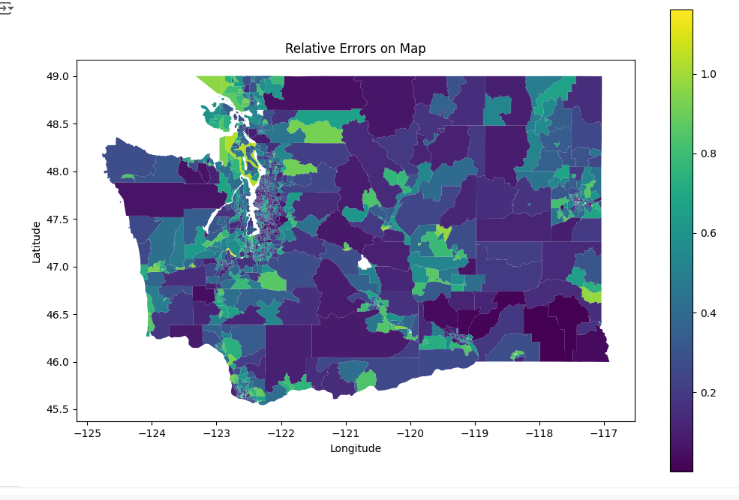
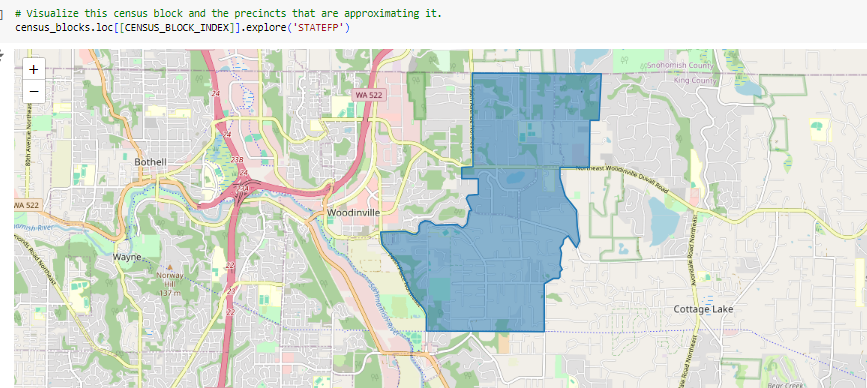
1. **Voting Records Dataset Review:** We accessed the voting records dataset via a link provided by Harvard University's database. This dataset was crucial as it contained detailed records of voting patterns, which are instrumental for analyzing trends and voter turnout in various demographics.
2. **Geological Dataset Review:** The geospatial dataset from the U.S. Census Bureau was reviewed to understand the geographical distribution and topographical influences on voting behavior. By accessing this dataset through the Census Bureau’s dedicated link, we were able to incorporate geographical data into our analysis, providing a spatial perspective to the voting patterns.
3. **Census Dataset Review**: Utilizing the U.S. Census Bureau's database, we explored various datasets, with a specific focus on income data. To find the most relevant information, we conducted a search for "INCOME" and applied filters to narrow down the data to "Census Tract" level for all the census tracts within Washington. We selected the year 2020 to ensure that our data reflected the most recent demographic and economic conditions.

The search results were meticulously examined to identify a table that contained useful features for our study. This table was critical as it allowed us to analyze how regional income levels influence voter behavior patterns in Washington state.

**Data Processing**

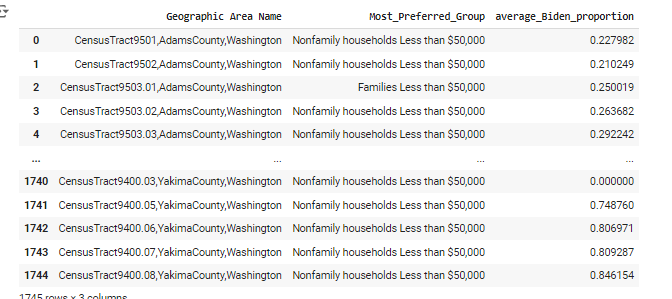
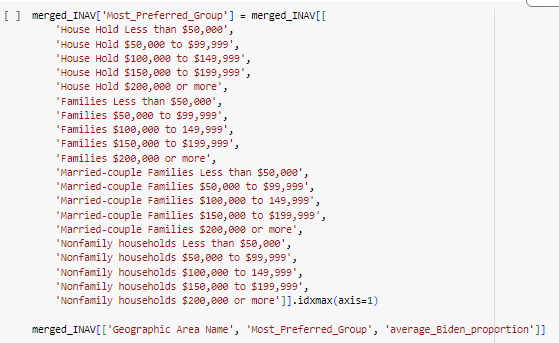
**Data Collection:** Following the method applied in the New York State analysis, this project extended similar geographical and electoral district analysis to Washington State using the file '240209 WA\_MatchTractsWithPrecincts\_CentroidMethod.ipynb'. Building upon the foundational work shared by a colleague from last year's capstone project, which used the '240202 NY\_YB\_MatchTractsWithPrecincts\_CentroidMethod', this approach was replicated to assess the demographic and electoral dynamics in Washington State. Utilizing GIS technology and the GeoPandas library, this file helped the calculation of geographic centroids for electoral districts in Washington, aligning them with their nearest voting precincts. This methodological transfer from the New York study to Washington ensured that the insights drawn could be compared across different regions, enhancing the robustness of the voting behavior analysis.

By leveraging previously validated methods, the study aimed to provide a comprehensive understanding of how electoral districts' demographic and economic variables affect voting patterns in Washington State, paralleling the analysis conducted for New York.



**Data Filtering:** The data filtering process was carried out in '240223Filtered Data.ipynb', focusing on efficiently extracting relevant information from the income data of Washington State. Initially, the data was scrutinized, and columns with unnecessary information or high rates of missing values were removed to clean the dataset. Missing values were either replaced or removed to ensure data accuracy. Additionally, data types were adjusted, and specific criteria were applied to filter data, aiming to prepare an optimal dataset for analysis. This refined dataset was ultimately used to prepare for model training and to analyze voter behavior predictions.

**Data Integration for Final Data:** The "240223Final Data.ipynb" file thoroughly describes the process of preparing the final dataset. Initially, it reads the "filtered\_income.csv" file to create a dataframe, which includes geographic area names and median income levels for households and families. The next step involves filtering Washington state data from the "bidenprop\_county.csv" file to examine Biden's support rates across major counties. This allows for the integration of electoral data with income data, setting a foundation to analyze how income levels correlate with voting behavior in each county. Subsequently, county names are added to the income data, and it is merged with the 'bidenprop\_county\_WA' dataframe. Finally, the data is saved to '240223Finaldata.csv' for further analysis. This procedure ensures meticulous data integration and cleaning, enabling more precise data analysis.



**Experiment and Evaluation**

1. **Simple Voting\_modeling: 240308 YB Voting\_modeling.ipynb**

Data was loaded using pd.read\_csv, and necessary preprocessing steps were undertaken to extract the County column and handle missing values. This data was then formatted to be provided in batches through PyTorch's Dataset and DataLoader classes, enabling efficient data handling for model training.

A simple multilayer perceptron model, named SimpleNN, was defined. It featured an input layer, a hidden layer with 32 neurons, and an output layer, utilizing the ReLU activation function for non-linear processing. This model was trained on preprocessed data, learning to approximate the relationship between the inputs and the voting outcomes. This model includes only one hidden layer with 32 neurons. This is a very basic setup, primarily suitable for less complex datasets where the relationship between the input features and the target is relatively straightforward. A custom loss function was defined to compute the Mean Squared Error (MSE) between the model’s outputs and the actual data labels. The model’s parameters were optimized using the Stochastic Gradient Descent (SGD) optimizer, which iteratively adjusted the weights to minimize the loss. The trained model was evaluated by applying it to test data and calculating the Root Mean Squared Error (RMSE). This evaluation metric helped to assess the model’s accuracy in predicting voting percentages across different counties.

1. **Improved Voting modeling: 240322 Improved Deep Voting model.ipynb**

The enhanced model, "DeepNN," expanded upon the limitations of "SimpleNN" by incorporating multiple hidden layers, thereby increasing the network's depth. Specifically, "DeepNN" introduced three additional hidden layers following the input layer, designed to enable the effective learning of more complex patterns and nonlinear relationships in the data. Techniques such as batch normalization and dropout were applied between these layers to stabilize the learning process and prevent overfitting. Batch normalization normalizes the inputs of each layer, making the training process faster and less sensitive to the initial weights, while dropout helps mitigate overfitting by randomly omitting some neurons during training, ensuring the model does not become overly reliant on specific neurons. These structural enhancements significantly improved the model's ability to generalize and learn intricate data structures compared to "SimpleNN."

1. **Different Model: 240322 Different ML.ipynb**

The "240322 Different ML.ipynb" file utilizes various machine learning models to analyze electoral data. The file includes loading the data, basic data processing, and then applying and evaluating several regression models.

* Linear regression employs a simple linear model as a straightforward method to establish a baseline for performance. It aims to clearly understand the characteristics that influence predictions, providing a foundational comparison point for more complex models. The performance of the linear regression model is evaluated using the Root Mean Square Error (RMSE), which quantifies the average magnitude of the prediction errors. In this case, the RMSE is calculated to be 0.1516, indicating the typical deviation of the predicted values from the actual values.
* Polynomial Regression with Ridge Regularization extends the capabilities of linear regression by incorporating polynomial features to capture more complex relationships within the data. This model also includes Ridge regularization, which is designed to prevent overfitting by penalizing larger coefficients in the model. The primary purpose of this approach is to determine whether increasing the complexity of the model leads to significant improvements in performance. The accuracy of the polynomial regression model is evaluated using the Root Mean Square Error (RMSE), enabling a direct comparison of how model complexity influences performance. In this instance, the RMSE is recorded at 0.1348, reflecting the model's enhanced ability to predict outcomes more accurately compared to simpler models.
* The Random Forest Regressor utilizes an ensemble of decision trees to enhance prediction accuracy by reducing variance. This method averages the outcomes from multiple decision trees to improve the reliability and accuracy of its predictions. The performance of this model is assessed using the Root Mean Square Error (RMSE), which measures the average magnitude of the prediction errors, indicating the model's accuracy. Despite its complexity and robustness against overfitting, there may be occasional performance inconsistencies. The model achieved an RMSE of 0.0797, indicating it had the most effective performance among the evaluated models.
* Support Vector Regression (SVR) adapts the principles of Support Vector Machines for regression tasks. It aims to fit the data within a specific error threshold, focusing on capturing complex nonlinear relationships in the data. The model's accuracy is evaluated using RMSE, which helps in assessing how closely the predicted values match the actual data points. SVR achieved an RMSE of 0.1213, demonstrating a reasonable level of accuracy, though not as high as the Random Forest Regressor.

1. **Different Evaluation: 240417 Different Eval.ipynb**

The "240417 Different Eval.ipynb" file is an adaptation of the "240322 Different ML.ipynb" base model, employing diversified evaluation methods. This file focuses on data handling techniques, particularly emphasizing model cross-validation and the use of various performance metrics to assess model effectiveness.

**Cross-validation:** The data is divided into multiple subgroups, with each subgroup used as a validation set to iteratively train and evaluate the model. This process assesses the model's ability to generalize, ensuring it is not overly optimized for specific subsets of data.

**Evaluation Metrics**

* RMSE (Root Mean Square Error) calculates the square root of the average squared prediction errors. This metric is used quantitatively to assess how well the model predicts.
* MAE (Mean Absolute Error) measures the average absolute error between predictions and actual values. This metric is useful for assessing the accuracy of predictions and is less affected by outliers.
* R² (R-squared, Coefficient of Determination) indicates how well the model explains the variability in the data. An R² value close to 1 indicates that the model effectively explains the data.

These evaluation methods help to comprehensively understand how accurately the model predicts real electoral data and assist in selecting the optimal model. Thus, the varied evaluation approaches provided in this file allow for an effective assessment of model performance, considering the complexity of the data.

**Result and Discussion**

This study evaluated the predictive power of various machine learning models on electoral data, focusing on identifying the strengths and weaknesses of each model to determine the best model for predicting election outcomes.

**Model Performance Comparison:**

* Random Forest: This model exhibited the lowest RMSE (0.0796), MAE (0.0584), and the highest R² (0.88), indicating its effectiveness in capturing and predicting the complex nonlinear interactions and patterns in electoral data. The high R² value suggests that the model explains a significant portion of the variability in the data.
* Linear Regression: As the most basic model, linear regression showed the highest RMSE (0.1515) and MAE (0.1203), with a relatively low R² value of 0.55, suggesting it fails to account adequately for the complexity in the data.
* Polynomial Regression with Ridge Regularization: Exhibited improved performance over linear regression with RMSE at 0.1348, MAE at 0.1021, and R² at 0.63. The addition of Ridge regularization helped mitigate overfitting, but the model still underperformed compared to the Random Forest.
* Support Vector Regression (SVR): Demonstrated moderate performance, with RMSE at 0.1213, MAE at 0.0902, and R² at 0.68. While SVR is effective in handling nonlinear data, it falls short in fully capturing the complexities of electoral predictions.
* The Random Forest model was identified as the most suitable for analyzing electoral data, attributed to its high predictive accuracy and ability to reflect the complexity of the data. In contrast, linear and polynomial regressions showed relatively lower performance, and SVR was only moderately effective.

**Proposed Solution for Lack of Individual-Level Data**

To address the challenge of not having individual-level voting data, we designed a new training method using a batch-average loss function. This function calculates the average prediction within a batch and compares it to the known target average for that batch. This method allows us to train the model without needing individual target values.

### **Construction of the Batch-Average Loss Function**

The batch-average loss function is constructed to focus on the mean predictions rather than individual predictions. Here's a simplified version of how it works:

1. For each batch, we calculate the average prediction from the model.
2. We then compare this average prediction to the known average target value for the batch.
3. The loss is computed as the mean squared error between these two averages.

This approach ensures that the model learns to predict the correct overall distribution of the target variable without needing precise individual target values.

### **Ensuring Replicability of Experiments**

### Data Preprocessing:

* Handling Missing Values: We used techniques such as mean imputation and forward filling to handle missing values in the dataset.
* Standardization: We standardized the columns to have a mean of 0 and a standard deviation of 1 to ensure that all features contribute equally to the model's learning process.
* Feature Selection: We reduced the number of features to a manageable number by selecting only those that showed significant variance or relevance to the target variable.

Model Architecture:

* Detailed descriptions of the model architecture, including the number of layers, the type of activation functions used, and any regularization techniques applied.
* Hyperparameters: We provided specific details on the hyperparameters used for training, such as learning rates, batch sizes, and the number of epochs.

Training Procedures:

* Explanation of the training loop, including how batches are processed and how the loss is calculated and minimized using backpropagation.
* The use of optimizers, including experiments with different optimizers like SGD and Adam, and the rationale for choosing the final optimizer based on performance.

Batch-Average Loss Function:

* Detailed construction and implementation of the batch-average loss function, including code snippets to illustrate how the function is applied during training.
* The logic behind comparing batch averages instead of individual predictions, ensuring the model can learn from aggregate data.

Code and Documentation:

* Providing code snippets and links to the relevant notebooks to allow others to follow along with the exact steps taken.
* Clear documentation within the code to explain each step, making it easier for others to understand and replicate the process.

By providing comprehensive details and clear documentation, we aim to make our experiments fully replicable by other researchers. This ensures that our findings can be validated and built upon in future studies.

However, the structure of the columns varied from state to state, making it a priority to standardize them. There were often inconsistencies in column names or formats, such as differences in spacing or terminology. To address this, we focused on standardizing the column names during the data preprocessing stage and ensuring consistent data formatting when necessary.

**Lesson**

* **GIS and Electoral District Data:** Through using GIS to adjust and analyze electoral district data, I realized the importance of handling geographic data. This experience will be incredibly beneficial for other projects that utilize geographic data.
* **Experimenting with Various Machine Learning Models:** This project provided me with my first opportunity to experiment with various machine learning models, allowing me to learn about the characteristics of each model and compare their performances. Understanding the importance of selecting the right model and finding the one that best fits the data was a crucial lesson, which will significantly aid in enhancing prediction accuracy.

**Challenges Faced**

When creating '240209 WA\_MatchTractsWithPrecincts\_CentroidMethod.ipynb' based on '240202 NY\_YB\_MatchTractsWithPrecincts\_CentroidMethod', I encountered several challenges. The main issue was the difference in column structures between the Washington State data and the New York State data. This necessitated editing and standardizing the columns. Initially, I chose California, but I had to abandon it due to the significant differences in the data.

In '240223Filtered Data.ipynb' and '240223Final Data.ipynb', the data was divided into categories such as married and unmarried, and there were numerous columns with mean and mode values. I had to reduce these columns to within 30. Additionally, the income data was divided into $10,000 intervals, making it difficult to analyze the society. To address this, I categorized the income into $50,000 intervals to enable clearer analysis.

When creating '240322 Different ML.ipynb', I experimented with various optimizers, including Adam. However, contrary to expectations, the originally used SGD yielded the highest accuracy. This was likely due to the fact that SGD, despite being simpler, is sometimes better suited for certain types of data and models where more complex optimizers might overfit or not converge as efficiently.

Furthermore, when initially creating the 'Improved Deep Voting model.ipynb', I attempted to add six hidden layers to SimpleNN. However, the results were not satisfactory because the model became too complex and suffered from overfitting. Therefore, I had to switch to using three hidden layers, which provided a better balance between complexity and generalization.

**Conclusion**   
This study presents a novel approach to voter behavior analysis, which can be applied to different types of electoral data or data from other states in the future. Additionally, integrating comprehensive data (such as age and education) and improving prediction accuracy through the integration of additional data sources or algorithm enhancements require further research.

**Reference**

Training Neural Networks with Batch Statistics